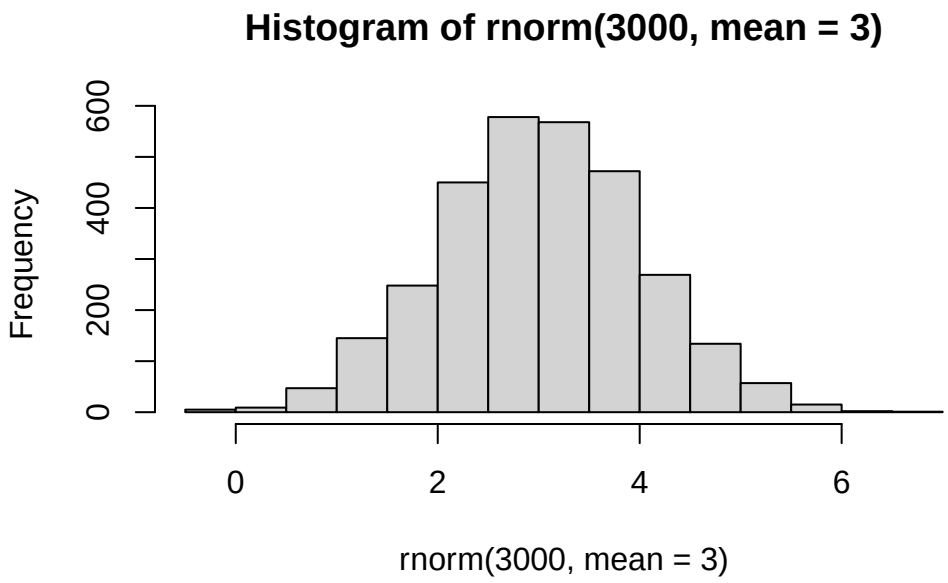
class07

Isabella Ruud (PID: A59016138

Today we will delve into unsupervised machine learning with an initial focus on clustering and dimensionality reduction.

let's start by making up some data to cluster: The `rnorm()` function can help here

```
hist(rnorm(3000, mean = 3))
```



Let's get some data centered at 3,-3 and -3,3

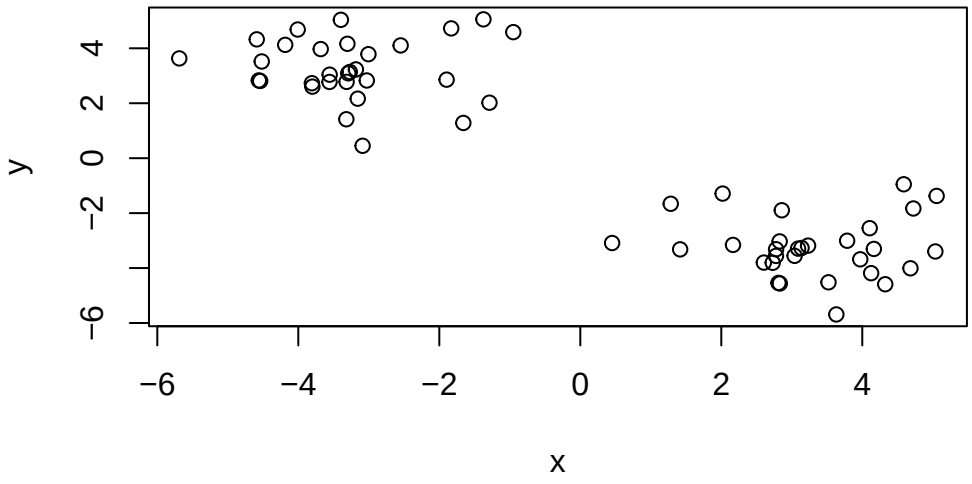
```
#combine 30 +3 values with 30 -3 values
x <- c(rnorm(30, mean = 3), rnorm(30, mean=-3))

#bind the values together columnwise
```

```r
z <- cbind(x=x, y=rev(x))
head(z)
```

```
             x         y
[1,] 2.8303160 -4.560473
[2,] 0.4513854 -3.087908
[3,] 2.7752559 -3.557454
[4,] 2.8278830 -3.028664
[5,] 3.5207117 -4.518548
[6,] 4.1050156 -2.548222
```

```r
plot(z)
```



##K means now we can see how k means clusters this data. the main function for k means clustering in base R is `kmeans()`

```r
km <- kmeans(z,centers = 2)
km
```

```
K-means clustering with 2 clusters of sizes 30, 30
```

```
Cluster means:
          x         y
1 -3.246631  3.258740
2  3.258740 -3.246631

Clustering vector:
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 71.53074 71.53074
 (between_SS / total_SS =  89.9 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

can list out the attributes and use the $ to access those components

```
attributes(km)
```

```
$names
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"

$class
[1] "kmeans"
```

What size is each cluster?

```
km$size
```

```
[1] 30 30
```

The cluster membership vector (ie the answer, or the cluster to which each point is allocated)

```
km$cluster
```

```
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

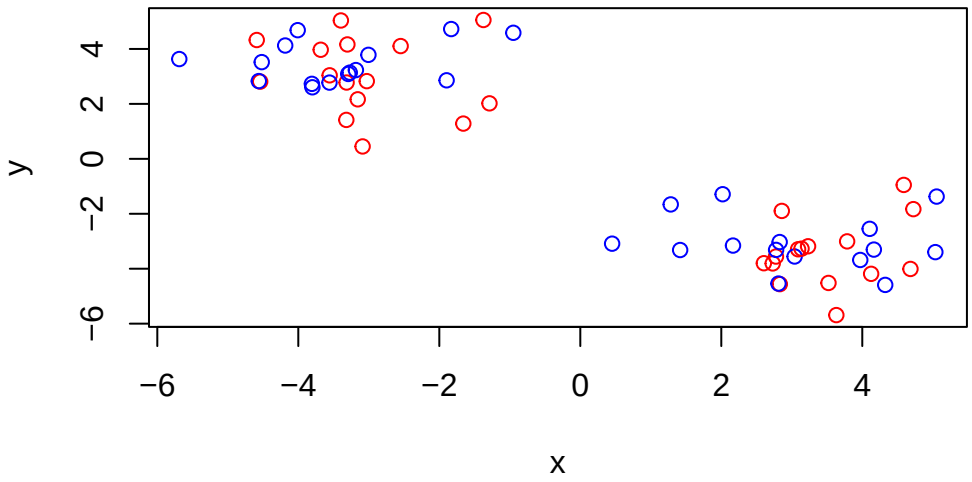
Cluster centers

```
km$center
```

```
          x         y
1 -3.246631  3.258740
2  3.258740 -3.246631
```

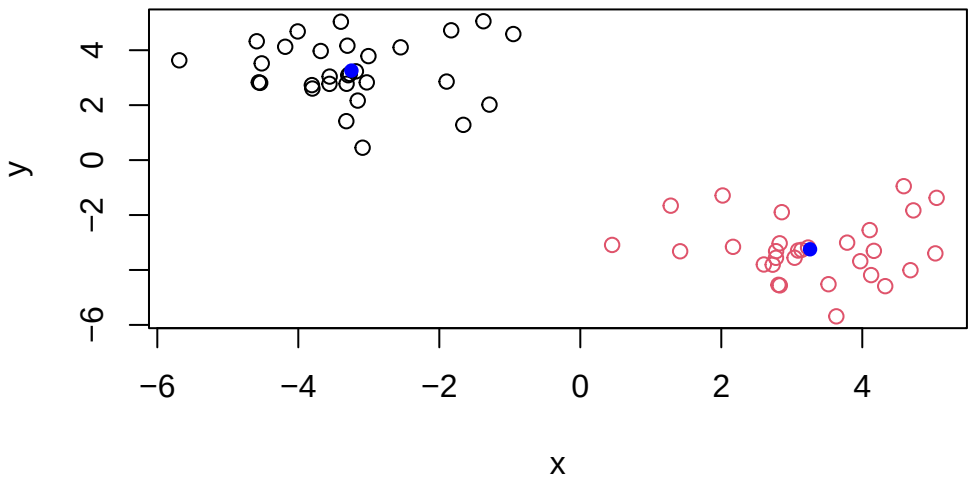
Make a results figure by plotting data colored by cluster membership and show the cluster centers

```
plot(z, col = c("red", "blue"))
```



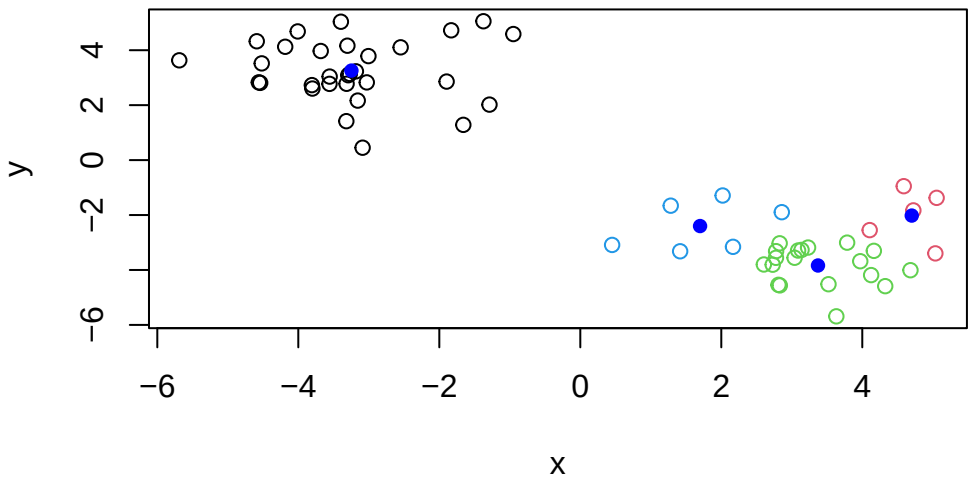
you can specify color based on a number, where 1 is black, 2 is red and you can use the cluster membership vector to color the points by cluster

```
plot(z, col=km$cluster)
points(km$centers, col = "blue", pch=16)
```



rerun your k means clustering using 4 clusters and plot the results as above

```
km_4 <- (kmeans(z,centers=4))
plot(z, col=km_4$cluster)
points(km_4$centers, col = "blue", pch=16)
```



Hierarchical clustering

The main base R function for this is `hclust()`. Unlike kmeans(), you can't just give your dataset as input, you need to provide a distance matrix.

we can use the `dist()` function for this

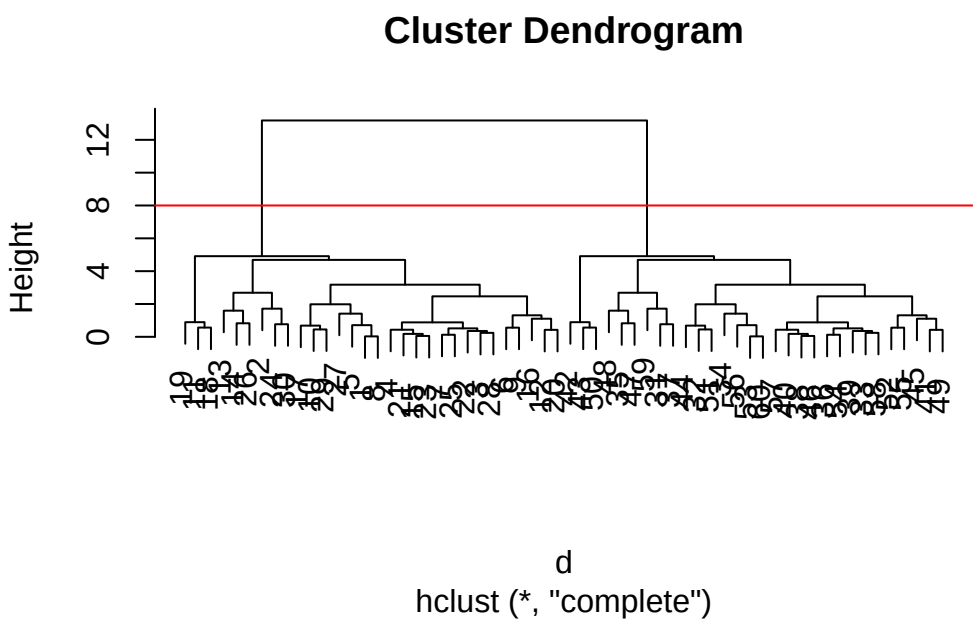
```
d <- dist(z)
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

Cluster method   : complete
Distance         : euclidean
Number of objects: 60
```

There is a custom plot() for hclust objects, let's see it

```
plot(hc)
abline(h=8, col="red")
```



The function to extract clusters/groups from your hclust object/tree is called `cutree()`

```
grps <- cutree(hc, h=8)
grps
```

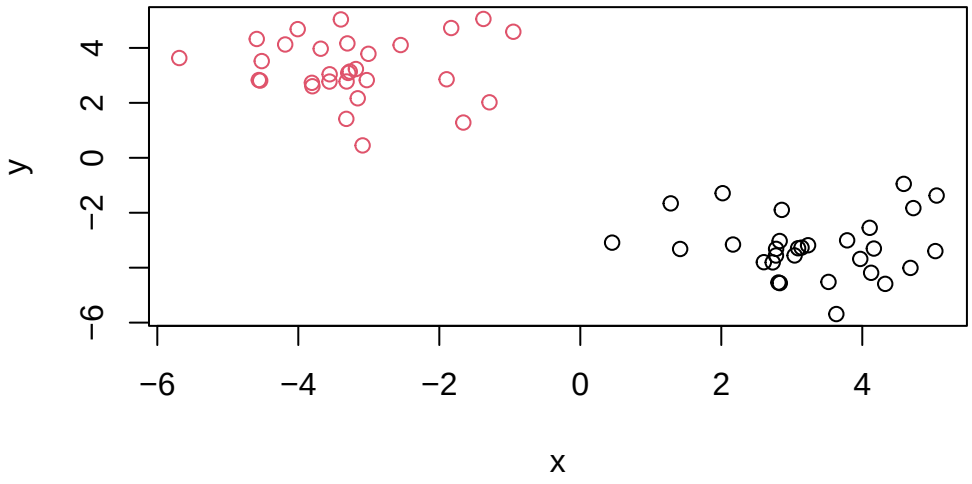
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
grps_k <- cutree(hc, k=2)
grps_k
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Plot data with hclust clusters

```
plot(z, col =grps)
```



Principal component analysis (PCA)

The main function for PCA in base R is `prcomp()` There are many add on packages with PCA functions tailored to particular data tpyes (RNAseq, protein structures, metagenomics etc)

PCA of UK food data

Read the data into R, it is a csv file and we can use `read.csv()` to read it

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
x
```

```
                   X England Wales Scotland N.Ireland
1             Cheese     105   103      103        66
2       Carcass_meat     245   227      242       267
3         Other_meat     685   803      750       586
4               Fish     147   160      122        93
```

```
5       Fats_and_oils      193   235      184       209
6               Sugars     156   175      147       139
7      Fresh_potatoes      720   874      566      1033
8           Fresh_Veg      253   265      171       143
9           Other_Veg      488   570      418       355
10 Processed_potatoes      198   203      220       187
11      Processed_Veg      360   365      337       334
12        Fresh_fruit     1102  1137      957       674
13            Cereals     1472  1582     1462      1494
14          Beverages       57    73       53        47
15        Soft_drinks     1374  1256     1572      1506
16   Alcoholic_drinks      375   475      458       135
17      Confectionery       54    64       62        41
```

```
dim(x)
```

```
[1] 17  5
```

```
head(x)
```

```
              X England Wales Scotland N.Ireland
1        Cheese     105   103      103        66
2  Carcass_meat     245   227      242       267
3    Other_meat     685   803      750       586
4          Fish     147   160      122        93
5 Fats_and_oils     193   235      184       209
6        Sugars     156   175      147       139
```

I would like the food names as row names not as their own column of data (currently the first column). I can fix this like so:

```
rownames(x) <- x[,1]
x <- x[,-1]
head(x)
```

```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
Carcass_meat      245   227      242       267
Other_meat        685   803      750       586
Fish              147   160      122        93
Fats_and_oils     193   235      184       209
Sugars            156   175      147       139
```

However, the above way can start deleting the first column if you keep running the code over and over again. A better way to do this is to do it at the time of data import with the read.csv() function.

```
url <- "https://tinyurl.com/UK-foods"
food <- read.csv(url, row.names = 1)
head(food)
```

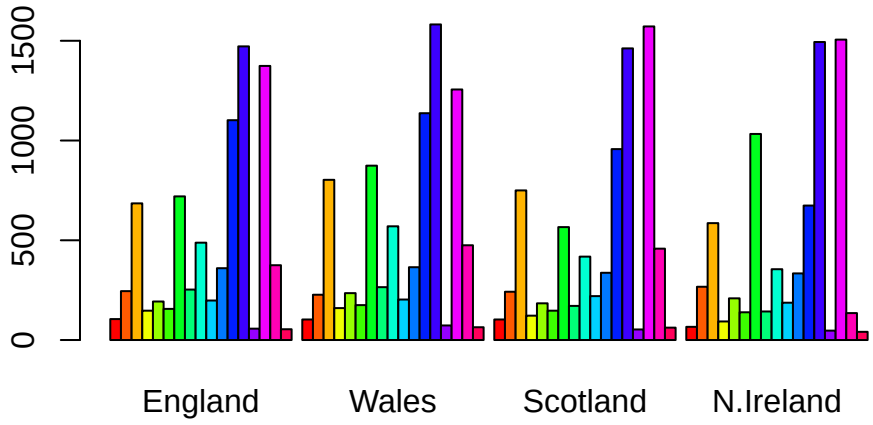
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars             156   175      147       139
```

Lets make some plots and dig into the data

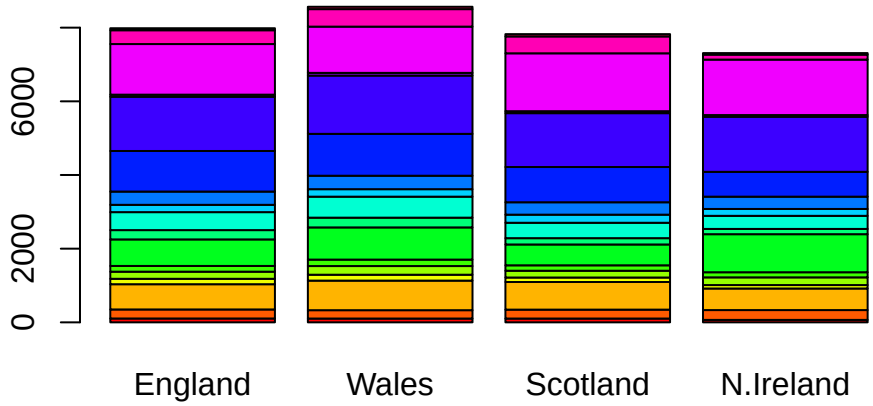
```
rainbow(nrow(food))
```

```
 [1] "#FF0000" "#FF5A00" "#FFB400" "#F0FF00" "#96FF00" "#3CFF00" "#00FF1E"
 [8] "#00FF78" "#00FFD2" "#00D2FF" "#0078FF" "#001EFF" "#3C00FF" "#9600FF"
[15] "#F000FF" "#FF00B4" "#FF005A"
```

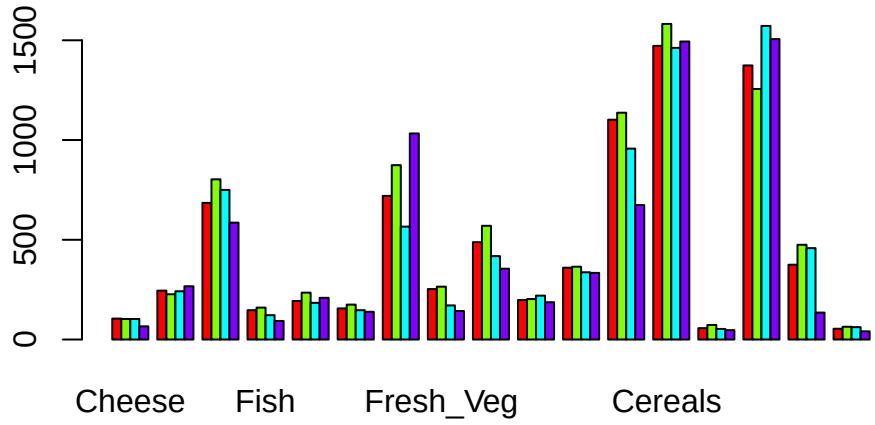
```
barplot(as.matrix(food), beside=T, col=rainbow(nrow(food)))
```




```
barplot(as.matrix(food), beside=F, col=rainbow(nrow(food)))
```

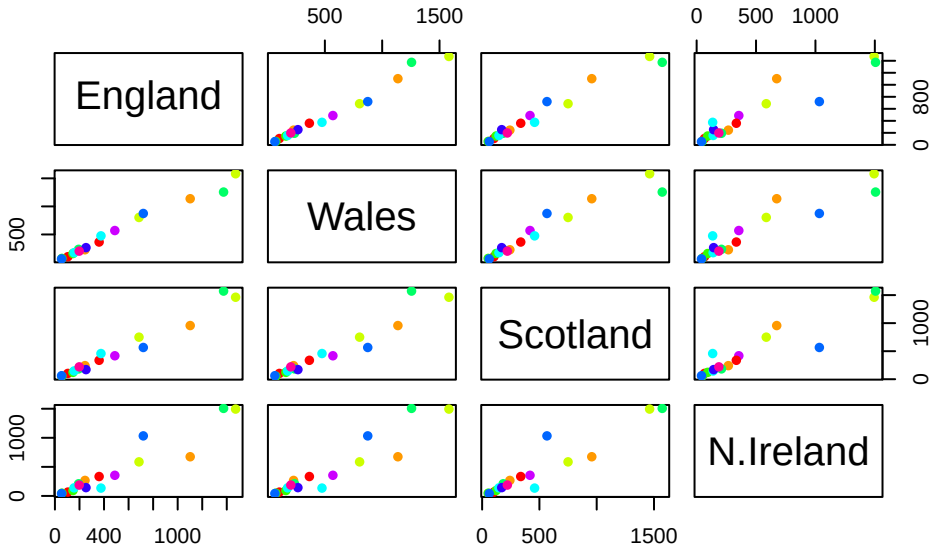


```
barplot(as.matrix(t(food)), beside=T, col=rainbow(nrow(t(food))))
```



How about a "pairs plot" where you plot each counry against all other countries. Dots that fall on the diagonal indicate that those values are similar between the two countries. We can see that other countries compared to North Ireland have two dots off the diagonal: higher potato consumption in North Ireland and less fresh fruit consumption in North Ireland

```
pairs(x, col=rainbow(10), pch=16)
```



##PCA A better way is to run a principal component analysis for this data using the prcomp() function. The prcomp() function will want the data to be transposed so that the countries are the rows and the food are the columns

```
pca <- prcomp(t(food))
summary(pca)
```

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

What is in my pca object?

```
pca
```

```
Standard deviations (1, .., p=4):
[1] 3.241502e+02 2.127478e+02 7.387622e+01 3.175833e-14

Rotation (n x k) = (17 x 4):
                              PC1           PC2          PC3           PC4
```

```
Cheese              -0.056955380  0.016012850  0.02394295 -0.694538519
Carcass_meat         0.047927628  0.013915823  0.06367111  0.489884628
Other_meat          -0.258916658 -0.015331138 -0.55384854  0.279023718
Fish                -0.084414983 -0.050754947  0.03906481 -0.008483145
Fats_and_oils       -0.005193623 -0.095388656 -0.12522257  0.076097502
Sugars              -0.037620983 -0.043021699 -0.03605745  0.034101334
Fresh_potatoes       0.401402060 -0.715017078 -0.20668248 -0.090972715
Fresh_Veg           -0.151849942 -0.144900268  0.21382237 -0.039901917
Other_Veg           -0.243593729 -0.225450923 -0.05332841  0.016719075
Processed_potatoes  -0.026886233  0.042850761 -0.07364902  0.030125166
Processed_Veg       -0.036488269 -0.045451802  0.05289191 -0.013969507
Fresh_fruit         -0.632640898 -0.177740743  0.40012865  0.184072217
Cereals             -0.047702858 -0.212599678 -0.35884921  0.191926714
Beverages           -0.026187756 -0.030560542 -0.04135860  0.004831876
Soft_drinks          0.232244140  0.555124311 -0.16942648  0.103508492
Alcoholic_drinks    -0.463968168  0.113536523 -0.49858320 -0.316290619
Confectionery       -0.029650201  0.005949921 -0.05232164  0.001847469
```

```
attributes(pca)
```

```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

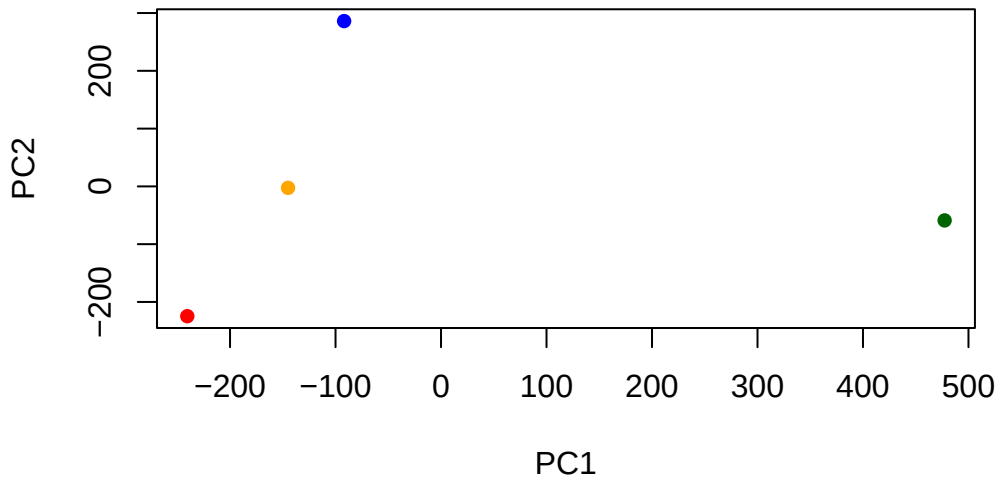
```
pca$x
```

```
                 PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

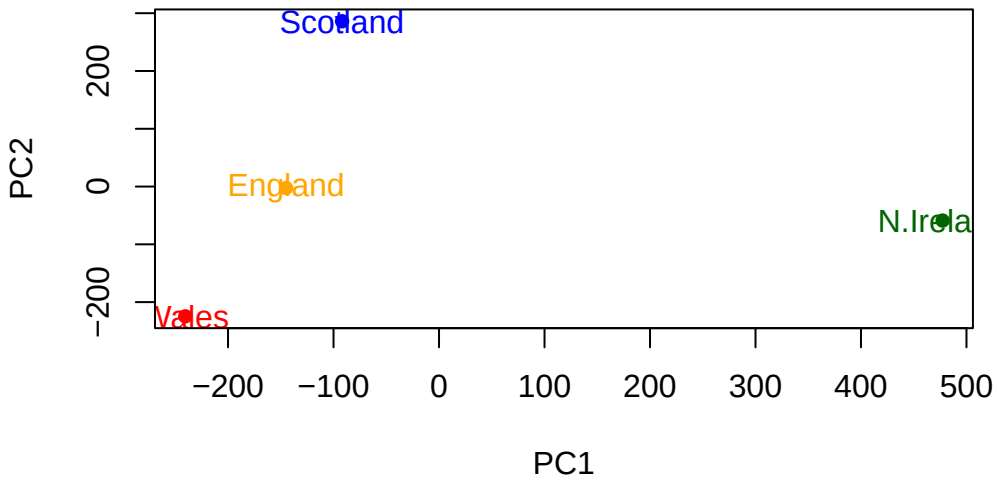
The scores are in pca$x

To make my main result figure, often called a PC plot, (or score plot or ordenation plot or PC1 vs PC2 plot)

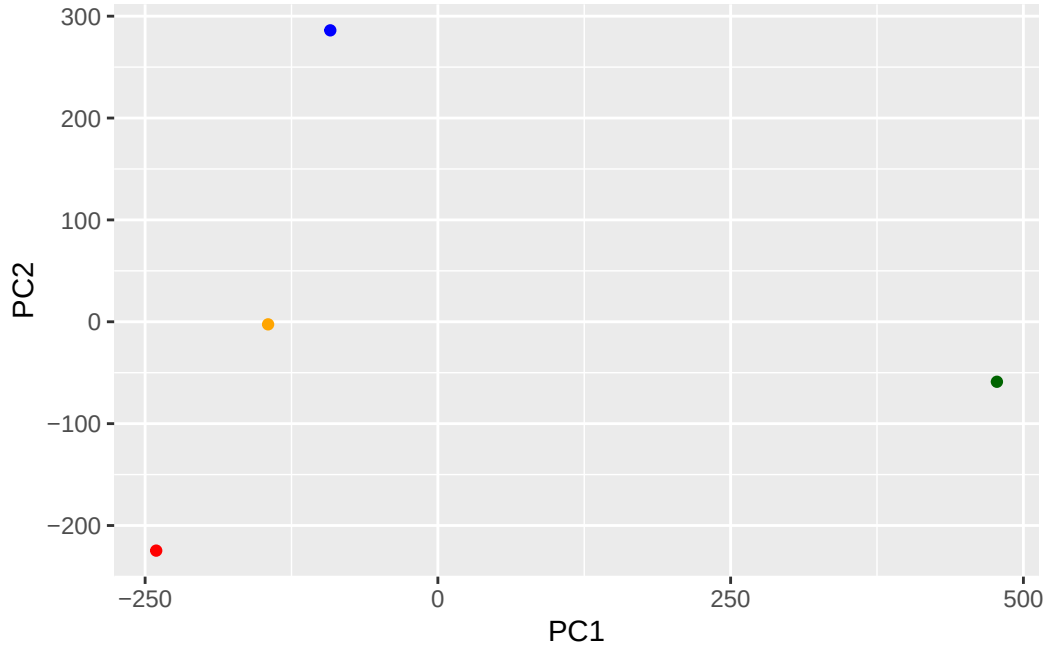
```r
plot(pca$x[,1], pca$x[,2], xlab = "PC1", ylab = "PC2",
     col = c("orange", "red", "blue", "darkgreen"), pch=16)
```




```r
plot(pca$x[,1], pca$x[,2], xlab = "PC1", ylab = "PC2",
     col = c("orange", "red", "blue", "darkgreen"), pch=16, text(pca$x[,1], pca$x[,2], colnar
```




```
library(ggplot2)
data <- as.data.frame(pca$x)
ggplot(data) + aes(x=PC1, y=PC2) + geom_point(col = c("orange", "red", "blue", "darkgreen"))
```



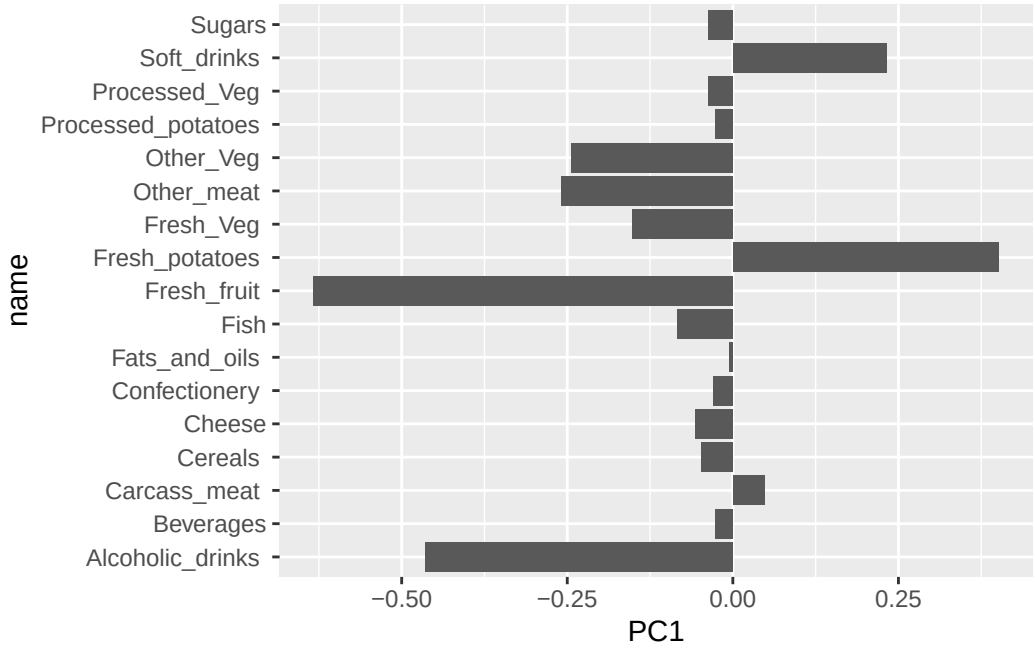
To see the contributions of the original variables (foods) to these new PCs, we can look at the pca$rotation component of the results

```
pca$rotation
```

```
                            PC1          PC2         PC3          PC4
Cheese               -0.056955380  0.016012850  0.02394295 -0.694538519
Carcass_meat          0.047927628  0.013915823  0.06367111  0.489884628
Other_meat           -0.258916658 -0.015331138 -0.55384854  0.279023718
Fish                 -0.084414983 -0.050754947  0.03906481 -0.008483145
Fats_and_oils        -0.005193623 -0.095388656 -0.12522257  0.076097502
Sugars               -0.037620983 -0.043021699 -0.03605745  0.034101334
Fresh_potatoes        0.401402060 -0.715017078 -0.20668248 -0.090972715
Fresh_Veg            -0.151849942 -0.144900268  0.21382237 -0.039901917
Other_Veg            -0.243593729 -0.225450923 -0.05332841  0.016719075
Processed_potatoes   -0.026886233  0.042850761 -0.07364902  0.030125166
Processed_Veg        -0.036488269 -0.045451802  0.05289191 -0.013969507
Fresh_fruit          -0.632640898 -0.177740743  0.40012865  0.184072217
Cereals              -0.047702858 -0.212599678 -0.35884921  0.191926714
Beverages            -0.026187756 -0.030560542 -0.04135860  0.004831876
Soft_drinks           0.232244140  0.555124311 -0.16942648  0.103508492
Alcoholic_drinks     -0.463968168  0.113536523 -0.49858320 -0.316290619
Confectionery        -0.029650201  0.005949921 -0.05232164  0.001847469
```

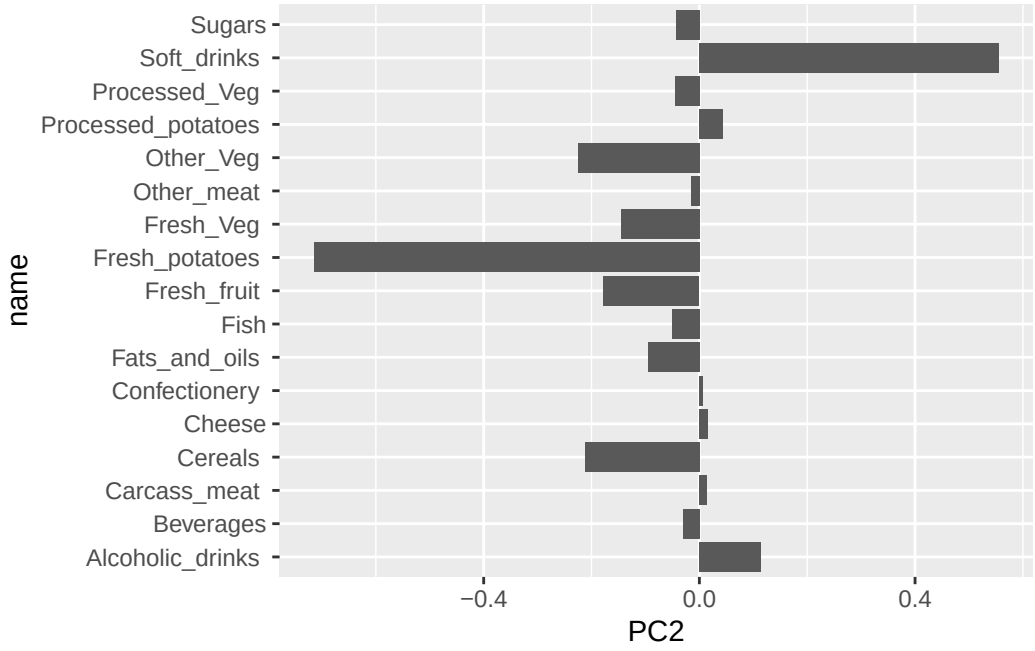
```
loadings <- as.data.frame(pca$rotation)
loadings$name <- rownames(loadings)

ggplot(loadings) + aes(PC1, name) + geom_col()
```




```
loadings <- as.data.frame(pca$rotation)
loadings$name <- rownames(loadings)

ggplot(loadings) + aes(PC2, name) + geom_col()
```


Sugars
Soft_drinks
Processed_Veg
Processed_potatoes
Other_Veg
Other_meat
Fresh_Veg
Fresh_potatoes
Fresh_fruit
Fish
Fats_and_oils
Confectionery
Cheese
Cereals
Carcass_meat
Beverages
Alcoholic_drinks
name
−0.4
0.0
0.4
PC2